Controlbot Final Report

(Previously known as Mazebot)

November 27, 2000

Group Members:

Steve Dillen	dillen@ee.ualberta.ca
Farrah Rashid	farrah@ualberta.ca

Declaration of Content

The design elements of this project and report are entirely the original work of the authors except as follows:

- H-Bridge Driver schematic taken from L298N data sheet
- Interface to Photo-reflector taken from the P5567 data sheet
- Fuzzy Controller is based on a design taken from EE564 Fall 2000 course notes from Dr. Pedrycz

______________________________ Farrah Rashid

______________________________ Steve Dillen

Abstract

Our project is to design a robot (Controlbot) that can control the speed of 2 motors in order to navigate in a straight line. Controlbot uses photo-reflective transister/infrared LED to encode data from each motor’s shaft encoder. This shaft encoder data is then converted to a Pulse Width Modulated (PWM) signal, which is applied to each motor. A fuzzy control algorithm is used to control the motors.

Achievements

Through the course of the project, Controlbot underwent some transformations, some disappointments, and some surprises. This is a brief list of what worked, and what didn't.

1) Name Change – Originally, the project was going to be a controlled mobile robot that would navigate through a maze. Unfortunately, this project was too large to be fully realized by the team, so Mazebot was downgraded to Controlbot.
2) Sonar – Unfortunately, the sonar devices we ordered were shipped to the wrong address (twice) so we didn't receive them in time for Controlbot to use them. However, we did manage to complete a VHDL design for interfacing to the sonar device to measure the distance from an object. This VHDL design synthesized and simulated correctly. Depending on the how well the rest of the project goes, the physical connection of the sonar's to the FPGA will be attempting to see how well the design works and if it can be added as a special feature.
3) Range-Finders – After not receiving the sonar's, we ordered some laser range finder's that would send out an infrared signal and measure the distance which could be transmitted back to the FPGA by clocking in 8 data bits. Again these devices were received fairly late into the project so they were not tested physically either, but the VHDL design simulated well.
4) Fuzzy Controller – This was a big component of the project when at first we didn't think it would be. The fuzzy controller was implemented using very simple membership functions, and only 2 inputs to provide 2 outputs. The fuzzy controller is responding properly to the data being sent to it, and is a good proof of concept for a hardware based fuzzy algorithm.
5) Synchronizer – This was a very simple design that was created almost immediately, and has been used extensively through the system. It is a circuit that was originally used to synchronize external signals with a generic parameter to specify the number of flip-flops to chain together. However, this circuit is used as a D flip-flop set to one level, and more importantly as a register for passing signals through pipe-line stages. If one signal needs to go through a pipeline that has 5 stages, it can be sent through this one module by setting the generic parameter to 5. This has proven to be extremely beneficial.
6) Maze Algorithm – The original Mazebot required a maze solving algorithm to be put in place. Since Mazebot was changed to Controlbot, this feature was removed. However, the algorithm was designed, simulated with test benches, and 90% completed.
7) Altera problems – A major time consuming problem that was found was with the lpm_counter module. This problem did not manifest itself in the simulations and testbenches but did show itself in the hardware realization. The problem was when using a lpm_counter with both the synchronous clear and the clock enable lines, the clear does not take effect unless the clock is enabled. This is not mentioned in the description for this module, and took a considerable amount of time to track down. While, this is not a design achievement, I do feel it is an excellent problem-solving achievement.

Table of Contents

DESCRIPTION OF OPERATION **6**

MOTOR CONTROL 6

Fuzzy Motor Controller [1] *6*

SHAFT ENCODERS 7

MOTOR DRIVER 7

CONTROLBOT DATASHEET **9**

SIZE **12**

EXPERIMENTATION AND CHARACTERIZATION **13**

REFERENCES **15**

Description of Operation

The basic design of Controlbot is shown in Figure 1 below:

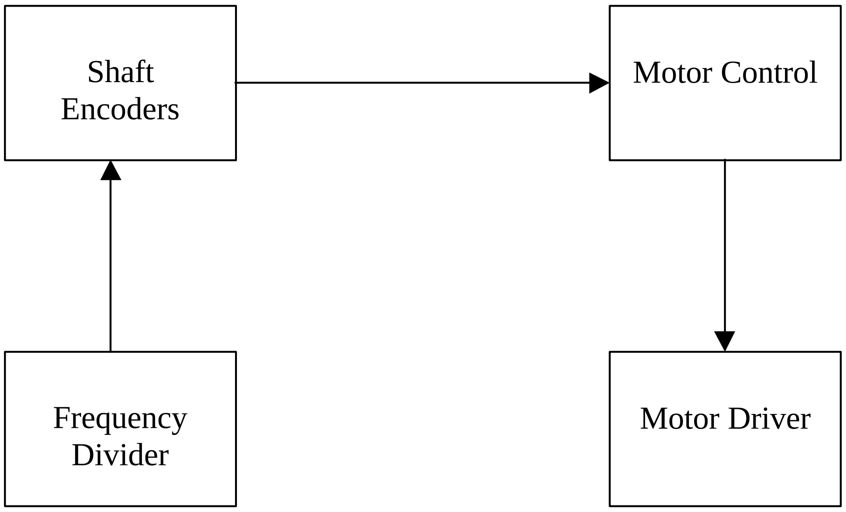


Figure 1: Block diagram for operation of Controlbot

Motor Control

The motor controller will use data received from the shaft encoders to control 2 DC motors using a fuzzy rulebase. The output of the motor control will be two PWM signals, which will be applied to the motors.

Fuzzy Motor Controller [1]

The fuzzy controller will use a rule-based inference engine to determine what speed to turn the motors. This controller will take 2 inputs to the rule base will be used to produce a single output. The inputs will be the difference in the velocity of the motor with respect to the reference velocity, and the change in this difference. This will implement a fuzzy PD controller. The output will be the value to change the duty rate for the motor. Figure 2 shows the block diagram of the fuzzy controller.

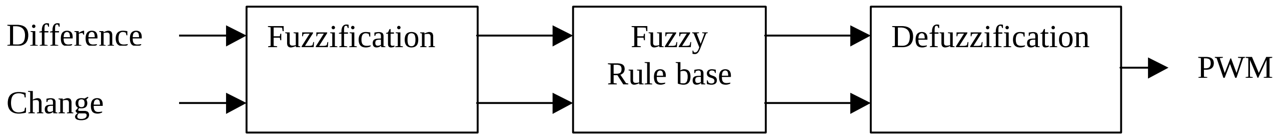


Figure 2: Fuzzy Rule-base

Since there are 2 motor’s being controlled, 2 fuzzy controllers have to be instantiated.

After experimenting with the fuzzy rule-base, it was decided that this algorithm is not the best for controlling 2 DC motors to the same velocity. An integral term integrating the difference between the left and right motor should be used as the second input rather than the change in difference to allow the motors to track each other. As it currently stands, the motors are completely independant of each other.

The fuzzy rule-base is implemented with the rules shown in Table 1.

Table 1: Fuzzy Rule base (Rows = Difference, Columns = Change in Difference)

	PL	***PM***	***PS***	***Z***	***NS***	***NM***	***NL***
PL	PL	PL	PL	PL	PL	PM	PM
PM	PL	PL	PM	PM	PM	PS	PS
PS	PS	PS	PS	Z	Z	PS	PS
Z	Z	Z	Z	Z	Z	Z	Z
NS	NS	NS	Z	Z	NS	NS	NS
NM	NL	NL	NM	NM	NM	NS	NL
NL	NL	NL	NL	NL	NL	NM	NM

Table 2: Legend

Abbreviation	***Meaning***
PL	Positive Large
PM	Positive Medium
PS	Positive Small
Z	Zero
NS	Negative Small
NM	Negative Medium
NL	Negative Large

Shaft Encoders

The shaft encoder is used to determine the actual speed of each wheel. This is done be applying a photoreflector to a disk consisting of black and white strips. When the photoreflector detects a white strip, the photo-transistor will turn on and send a logic ‘1’ to the FPGA. When a black strip is detected, the transistor will turn off and send a logic ‘0’ to the FPGA.

Motor Driver

The output from the motor controller will be a Pulse Width Modulated (PWM) signal with the duty cycle dependant on the optical encoder data the motor controller received. The motor will be connected to its supply voltage through a switch controlled by the FPGA. The FPGA will apply voltage to the motor and remove it based on the PWM signal to achieve the desired average voltage. Figure 3 illustrates this relationship.

Notice in Figure 3 that the PWM signal does not go directly to the motor, but instead to an H-Bridge power transister chip. This is placed there so the FPGA board is separated from the motors, and the power transistors can source and sink the current required to drive the motors.

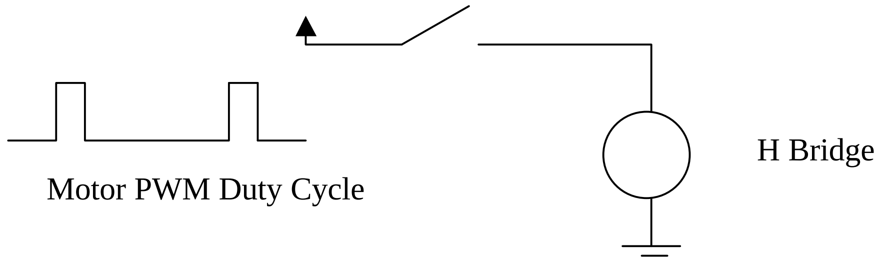


Figure 3: PWM signal

Controlbot Datasheet **CB001**

DC MOTOR CONTROLLED MOBILE ROBOT

- Operating supply Voltage up to 9 V DC
- TTL Level Outputs for H-Bridge Motor Control Interface
- Separate Power Supply for Motor and Chip (Common Ground)
- Fuzzy based Control Algorithm
- Easy configurable control parameters
- Internally synchronizes asynchronous inputs
- Maximum Clock Speed of 37 MHz
- 772 of total 1152 Logic Cells

DESCRIPTION

The CB001 is a DC Motor Control FPGA designed specifically for use with Mobile robots and maintaining a constant velocity. It requires interface circuitry for the motor to convert TTL level inputs to motor driver signals. The recommended chip for this is the L298 Dual Full-bridge Driver. Inputs required for this circuit is the actual velocity of the motor. The recommended method for this is to use a Hamamatsu P5585 Photoreflector in combination with a shaft encoder.

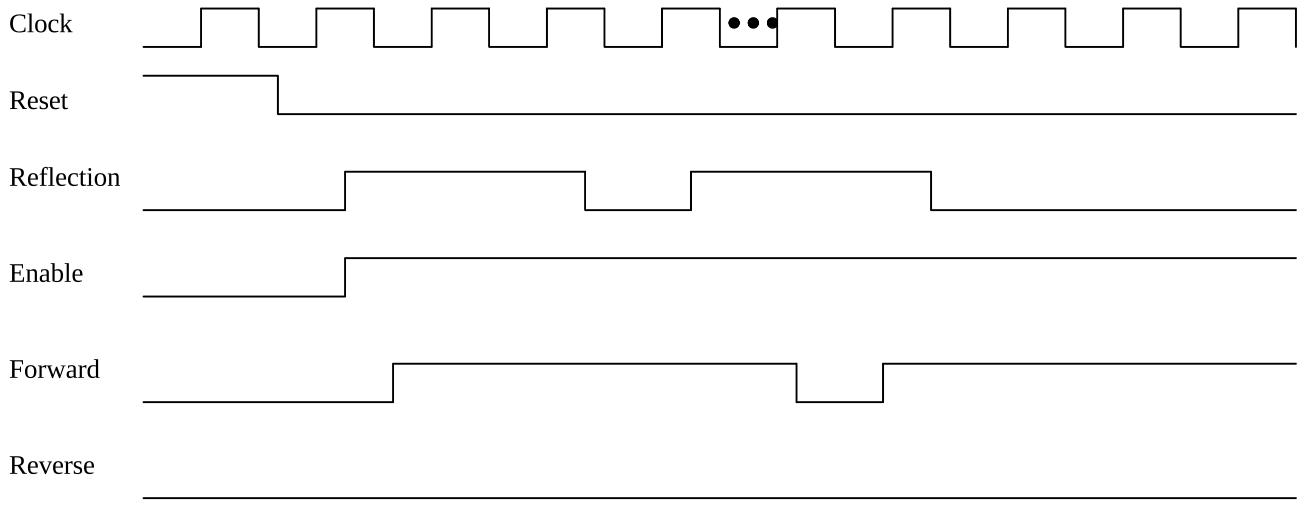
FEATURES

The internal Fuzzy controller encodes the incoming data and outgoing control change into 7 different fuzzy linguistics. They are: Positive Large, Positive Medium, Positive Small, Zero, Negative Small, Negative Medium, Negative Large. By maintaining an internal change in difference from the reference velocity, this provides a 49 rule fuzzy rulebase for controlling a motor.

PIN DESCRIPTIONS

Name	***Type***	***Function***
Clock	Input	Oscillator (Maximum 37 MHz)
Reset	Input	Active High reset line. System must be held in reset for at least 1 full clock period for proper operation
Enable0; Enable1	Output	Motor 0 and 1 enable lines.
Forward0; Forward1	Output	Motor 0 and 1 forward lines. These will drive the motors forward
Reverse0; Reverse1	Output	Motor 0 and 1 reverse lines. (Reserved for future use)
Reflection0; Reflection1	Input	Motor 0 and 1 reflection signals. The CB001 counts rising edges of the reflection line to determine control

EXAMPLE TIMING DIAGRAM



Size

Table 3 outlines the size and speed of the coded modules

Table 3: Current Logic Size and Speed

Module	***Size of Module (Logic Cells)***	***Maximum Speed (MHz)***
Controlbot	772	38.61
Motor Control	474	29.32
Controller	198	29.58
Input	21	79.36
Fuzzification	74	45.04
Fuzzy Rule base	61	40.98
Defuzzification	85	125
Duty Rate Generator	15	96.15
Latch Rate	31	120.48
PWM Generator	26	99
Duty Splitter	6	62.5
Duty Counter	4	125
Data Fusion	36	96.15
Motor driver	3	76.3
Shaft encoder	15	97.08
Photo reflector	2	125
Controlbot DFF	1	125
Controlbot mux2	2	76.3
Controlbot counter	2	125
Frequency divider	10	81.3
Preset Synchronizer	2	125
Synchronizer	2	125

Controlbot was compiled with a bus width of 12 data bits. All the other modules used a bus width of only 4 data bits in order to speed up simulation times. Because of this, the size numbers are low, and the frequency is high. This is all right so long as it is understood that these numbers are dependent on the bus width used.

Even with the bus width being set low, the bottlenecks in the design can still be isolated. The fuzzy rule base comprises a large number of cells and is the slowest module. This would be a good candidate for optimization. Currently, the fuzzy rule-base is implemented as a 2 stage pipeline. The first stage compares 7 sets of 7 rules. The next stage combines these rules into a single output. Once the optimal rulebase is found, this module can be modified to combinational logic using 6 variable karnaugh maps. This was not done since tuning of the rules is easiest when they are broken out into the more readable if/then form.

Due to the size of controlbot, once the motion control for wall detection and turning were in, the algorithm would not have fit on the chip. Controlbot commands almost 70% of the logic cells just for controlling the motors.

Experimentation and Characterization

A design was made for both the sonar sensor circuit and the shaft encoder. Each module required a resolution counter (1ms) and then a multiply or divide by a constant so a single calculator module could be used to calculate the distance or velocity. This circuit was designed and synthesized and found to work well. Unfortunately, it could only run at 4 MHz maximum clock according to the timing analyzer. Increase the pipelining to 2, 4, and 8 levels did not help the calculater, the best it could do was 8 MHz.

After realizing that divider's really aren't that optimal, we noticed that since both are just multiplying constants or dividing constants, we can remove the divider completely and change the control rules by dividing or multiplying the constant across.

The fuzzy rule-base was simulated in Matlab first to see if the rule-base would cause the motor's to converge. The response graph to this analysis is shown in Figure 4.

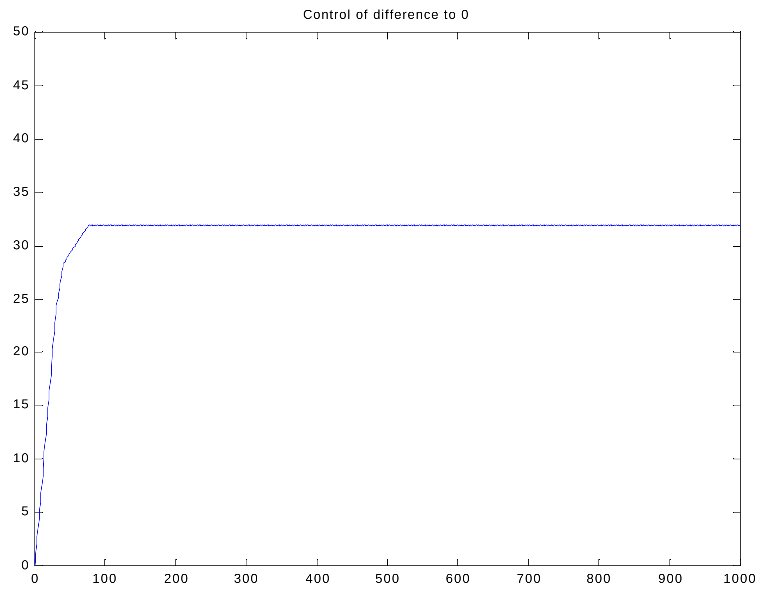


Figure 4: Simulated Response time

Matlab code can be found in Appendix D.

The largest problem with Controlbot as it stands is the resolution of control. A different shaft encoder was tried that had less resolution in terms of degrees (every 9 degrees instead of 15), but it did not help. A different implementation for the shaft encoder was tried that counted shaft encoder ticks (rising pulse edges), rather than timing the number of milliseconds between pulses. This resulted in a smaller, faster shaft encoder, but the control algorithm deteriorated. It was decided that the original shaft encoder would work the best.

References

1) EE564 Course Notes Fall 2000 from W. Pedrycz [1]
2) The Robot Builder’s Bonanza – 99 Inexpensive Robotics Projects by Gordon McComb 1987
3) Sensors for Mobile Robots Theory and Application by H.R. Everett 1995
4) Mobile Robots – Inspiration to Implementation by Joseph L. Jones 1993